Exercice 4.
Matrix manipulation.

Create the script "exercise4.R" and save it to the "Rintro/day2" directory: you will save all the commands of exercise 4 in that script.
Remember you can comment the code using #.

1. Create three numeric vectors **x, y, z,** each of 4 elements of your choice.
Use **rbind()** to combine the three vectors to become the matrix **mat** (3 rows and 4 columns) where each row represents a vector.

2. Create the same matrix with a different approach: the **matrix()** function.

3. Change the column names of **mat** to "a", "b", "c", "d".
colnames()

4. Calculate the sum of each row, and the sum of each column.
rowSums(); colSums()

5. Create the matrix **mat2**:
mat2 <- matrix(c(seq(from=1, to=10, by=2), 5:1, rep(x=2017, times=5)), ncol=3)

What does function **seq**() do?

6. What are the dimensions of **mat2** (number of rows and number of columns)?
dim(); nrow(); ncol()

7. Change the column names of **mat2** to "day", "month" and "year".
colnames()

8. Change the column names of **mat2** from "A" to "E"**.**
rownames()

9. Shows rows of **mat2** if the month column is superior or equal to 3.

10. Replace all elements of **mat2** that are equal to 2017 with 2018.

11. Multiply all elements of the **2d column** of **mat2** by 7. Reassign.

12. Add one column "time" to **mat2** that contains values 8, 12, 11, 10, 8.
Save in the new object **mat3**.

13. Replace all elements of **mat3** that are inferior to 3 by **NA**.

14. Remove rows from **mat3** if a **NA** is present. Save in **mat4**.
na.omit(); na.exclude()

15. Retrieve the minimum value of each column.
Try different approaches:
- Retrieve the minimum for each column separately.
- Retrieve the minimum of all columns simultaneously using the **apply()** function.

Exercise 5.
Data frame manipulation.

Create the script "exercise5.R" and save it to the "Rintro/day2" directory: you will save all the commands of exercise 5 in that script.
Remember you can comment the code using #.

1. Create the following data frame **df**:

	Age	Height	Sex
John	43	181	M
Jessica	34	172	F
Steve	22	189	M
Rachel	27	167	F

Row names are John, Jessica, Steve, Rachel.
Column names are Age, Height, Sex.

2. Check the structure of **df** with **str().**

3. What are the average age and height in this table?
Try different approaches:
- Calculate the average for each column separately.
- Calculate the average of both columns simultaneously using the **apply()** function.

4. Add one entry to **df**: Georges who is 53 years old and 168 tall.

5. Change the row names of **df** so the data becomes anonymous: use Patient1, Patient2, etc.

6. Create the data frame **df2** that is a subset of **df**: it will contain only the female entries.

7. Create the data frame **df3** that is a subset of **df**: it will contain only entries of males **taller than 170**.

Exercise 5b - OPTIONAL

1. Create two data frames **mydf1** and **mydf2.**
Column names are respectively "id", "age" and "id", "name".

id	age
1	14
2	12
3	15
4	10

id	name
1	paul
2	helen
3	emily
4	john
5	mark

2. Merge **mydf1** and **mydf2** by their "id" column. Save in **mydf3**.
merge()

3. Sort **mydf3** by decreasing age with the **order()** function.

Exercise 5c – OPTIONAL

1. Copy file **"/nfs/users/bi/sbonnin/genes_dataframe.RData**" in your current directory with the **file.copy()** function.

2. The function **dir()** lists the files and directories present in the current directory: check if "**genes_dataframe.RData**" was copied.

3. Load the "**genes_dataframe.RData**" file in your environment:
load()

4. "**genes_dataframe.RData**" contains the **df_genes** object: is it now present in your environment?

5. Check some characteristics of **df_genes** and see what it contains: **str(), head(), dim(), colnames(),** etc.

6. Select rows for which **pvalue_KOvsWT < 0.05 AND log2FoldChange_KOvsWT is > 0.5**. Store in the **up** object.
How many rows (genes) were selected?

7. Select from the up object the "Zinc finger protein" coding genes (i.e. the gene symbol starts with Zfp). Use the **grep()** function. How many genes do you get?

8. Select rows for which **pvalue_KOvsWT < 0.05 AND log2FoldChange_KOvsWT is > 0.5 OR < -0.5.** What about giving the **abs()** function a try? Store in the **diff_genes** object. How many rows (genes) were selected?